|  |
| --- |
| Document de suivi  de l’unité logicielle ULx  ~  Spécification de l’unité de programme  ~  Fiche de test pour validation de l’unité  ~  Bilan de validation  ~  Annexes  (sources, données de test, résultats…)  ~  Code Interne : SPEC-FDT-BV\_ULx |

Résumé

|  |  |  |  |
| --- | --- | --- | --- |
| * Présentation des spécifications fonctionnelles du composant * Présentation des spécifications d’interface du composant * Plan de validation du composant * Fiche de test type * Compte rendu de validation et Visa de validation |  | **Niveau de diffusion** | |
|  | Contrôlée |  |
|  | Interne | X |
|  | Libre |  |

Validation

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nom | Entité | Date |
| **Auteur(s)** | Jérémy Guimberteau | **L3 Miage –** Groupe 5 | 24/05/2009 |
| **Vérificateur(s)** | Anaïs Dubourg | **L3 Miage –** Groupe 5 | 25/05/2009 |
|  |  | ??/??/2009 |
| **Validation(s)** |  | **L3 Miage - Enseignant** | … |

Historique de mise à jour

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Rédacteur | Date | Objet | Paragraphe(s)  Concerné(s) |
| **S0F0** | Jérémy Guimberteau | 24/05/2009 | Création du document | De §‎1 à §‎5 |
| **S0F1** | Jérémy Guimberteau | 25/05/2009 | Bilan – Visa de validation | De §‎1 à §‎5 |
|  |  |  |  |  |

Documents associés

|  |  |
| --- | --- |
| Titre | Référence du fichier |
|  |  |

**Table des matières**

[1. Spécifications de l’unité de programme 4](#_Toc231661088)

[1.1 Contexte 4](#_Toc231661089)

[1.2 Spécifications fonctionnelles de l’unité logicielle 4](#_Toc231661090)

[1.2.1 Exigences fonctionnelles 4](#_Toc231661091)

[1.2.2 Performances et exigences qualité 4](#_Toc231661092)

[1.3 Spécifications d’interface de l’unité logicielle 5](#_Toc231661093)

[1.3.1 Description détaillée des interfaces 5](#_Toc231661094)

[1.3.2 Schéma de la solution retenue 5](#_Toc231661095)

[1.4 Contraintes de réalisation 5](#_Toc231661096)

[1.4.1 Conditions d’activation 5](#_Toc231661097)

[1.4.2 Conditions d’exception et d’arrêt 5](#_Toc231661098)

[2. Plan de validation de l’unité logicielle 6](#_Toc231661099)

[2.1 Stratégie 6](#_Toc231661100)

[2.2 Exemple de scénario valide 6](#_Toc231661101)

[3. Fiche de test pour la validation de l’unité 7](#_Toc231661102)

[4. Bilan de validation 8](#_Toc231661103)

[4.1 Validation par le programmeur 8](#_Toc231661104)

[4.2 Validation par des testeurs indépendants 8](#_Toc231661105)

[4.3 Visa de validation 8](#_Toc231661106)

[5. Annexes : 9](#_Toc231661107)

[5.1 Les sources du programme 9](#_Toc231661108)

[5.1.1 Chemin de l’arborescence\testTraitFormat.java 9](#_Toc231661109)

[5.1.2 Les fichiers d’entrée et de sorties 9](#_Toc231661110)

# Spécifications de l’unité de programme

## Contexte

L’unité logicielle AutoTest s’inscrit dans le cycle de développement du service d’autotest du composant de vérification syntaxique.

## Spécifications fonctionnelles de l’unité logicielle

### Exigences fonctionnelles

L’unité logicielle AutoTest a pour objectif de tester le service de vérification syntaxique.

Les spécifications de l’unité logicielle peuvent se résumer de la manière suivante :

1. Envoie des données au composant à tester
2. Récupère les données en sorties du composant et les écrit dans un fichier

L’unité logicielle AutoTest doit rendre possible l’**automatisation des tests** du composant final. Ainsi, il devra être possible de prendre en entrée un **fichier de jeu de données** et de produire en sortie un **fichier résultat** mémorisant les traitements effectués.

### Performances et exigences qualité

RAS

## Spécifications d’interface de l’unité logicielle

### Description détaillée des interfaces

L’unité logicielle **AutoTest** a une unique méthode publique : testTraitFormat(*String fichierJDT, final String fichierRUN*).

Le service de vérification syntaxique en cours de test sera présent au sein d’une autre sous-unité logicielle : **Stub.** L’automate de traitement traitFormat() n’étant pas encore programmé, la sous-unité logicielle est à un « Stub » : la méthode publique : Stub(). Cette méthode renverra un objet de type Format qui représente le format reconnu associé à la chaîne analysée.

### Schéma de la solution retenue

Voici le schéma UML représentant la solution retenue pour l’unité logicielle AutoTest :

|  |
| --- |
| AutoTest |
|  |
| public static void testTraitFormat(String fichierJdt, String fichierRun) |

## Contraintes de réalisation

### Conditions d’activation

L’unité logicielle **Autotest** a besoin d’un fichier en entrée contenant les données de test. Cela nécessite que le format de donnée soit correctement défini. Toute modification devra faire l’objet d’une demande justifiée qui sera exposée lors de la réunion hebdomadaire.

### Conditions d’exception et d’arrêt

L’unité logicielle ayant besoin d’un fichier « jeu de données de test » en entrée, si ce dernier n’est pas trouvé, le traitement prend fin. De la même manière, si l’écriture des résultats ne peut se faire dans un fichier valide, l’exécution de la méthode TestTraitFormat(String fichierJdt, String fichierRun) sera interrompue et l’unité logicielle ne pourra être considérée comme opérationnelle.

# Plan de validation de l’unité logicielle

## Stratégie

Pour pouvoir affirmer que l’unité logicielle AutoTest est 0-défaut, il est nécessaire que les résultats des tests effectués sur cette unité logiciel soient positifs.

L’unité logicielle sera valide si les fichier JDT.ref et JDT.run sont identiques

Voici un exemple de scénario permettant de valider l’unité logicielle.

## Exemple de scénario valide

**Soit le JDT suivant présenté en entrée :**

1 00

2 0000

3 00 00

4 99

**Le résultat attendu en sortie devra être :**

1 F1 0

2 F2 0 0

3 F3 0 0

4 F0 153

**Le fichier de référence :**

1 F1 0

2 F2 0 0

3 F3 0 0

4 F0 153

Maintenant que nous avons élaboré la stratégie de validation, nous pouvons définir une fiche de test type.

# Fiche de test pour la validation de l’unité

|  |  |  |  |
| --- | --- | --- | --- |
| **FICHE DE TEST** | | | |
| **Date de création de la FDT :** | 04/05/2008 | **Rédacteur :** | Jérémy Guimberteau |
| **Contact :** | jeremy.guimberteau@gmail.com | **Entité :** | **L3 Miage Paris Descartes** |
| **Version de la FDT :** | 1 | **Code interne** | **FDT\_AutoTest** |
| **Intitulé :** | **Fiche de test unitaire de l’unité logicielle AutoTest** | | |

|  |  |  |
| --- | --- | --- |
| **DESCRIPTIF** | | |
| **Choisir la typologie du test** | **Test Unitaire** | Test d’intégration |
| **Description du test  - Données en entrée - Données en sortie** | | |
| Ce programme va prendre en entrée un **fichier « jeu de données »** et va produire, en sortie, un **fichier résultat** mémorisant les traitements effectués.  1 F1 0  2 F2 0 0  3 F3 0 0  4 F0 153  **Données en entrée**  1 00  2 00 00  3 0000  4 99  Les données en entrées …. Voici un exemple de JDT que l’on pourra tester :  **Données en sortie**  Les données en sortie …. Voici un exemple de fichier *run* que l’on pourrait obtenir : | | |

|  |  |  |
| --- | --- | --- |
| **PROCEDURES DE TESTS ET D’EVALUTAION** | | |
| **Mode d’évaluation des résultats :** | **Humain** | Automatique |
| **Procédures de tests :** | | |
| Pour tester la méthode **testTraitFormat(String fichierEntree, String fichierSortie),** …  Les données de tests sont présentes dans le répertoire :  Data tests\TestTraitFormat.jdt (ex : fichier TestTraitFormat.jdt)  Les résultats de sorties seront écrits dans le répertoire :  Data test\JDT.run (ex : fichier JDT.run)  Pour l’évaluation des résultats et la validation de l’unité logicielle, … | | |

# Bilan de validation

## Validation par le programmeur

|  |  |  |  |
| --- | --- | --- | --- |
| **Date de passage du test :** | 25/05/2009 | **Rédacteur :** | Jérémy Guimberteau |
| **Testeur :** | Jérémy Guimberteau | jeremy.guimberteau@gmail.com | |
| **Compte Rendu de validation** | **La validation de l’unité logicielle AutoTest est acceptée.** | | |
|  | | | |

## Validation par des testeurs indépendants

|  |  |  |  |
| --- | --- | --- | --- |
| **Date de passage du test :** | 10/05/2008 | **Rédacteur :** | Jérémy Guimberteau |
| **Testeur(s) :**  (indépendant du programmeur) | Anais Dubourg | anais.dubourg@gmail.com | |
| **Compte Rendu de validation** | **L’unité logicielle est donc 0-Défaut.** | | |
|  | | | |

## Visa de validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Validation par le programmeur** | | | |
| **Visa Programmeur** | OK | **Date de validation** | 09/05/2008 |
| **Visa Testeur(s) indépendants** | OK | **Date de validation** | 10/05/2008 |
| **Conclusion** | **0-Défaut** | Bug Constaté | |
|  | | | |

# Annexes :

## Les sources du programme

### Chemin de l’arborescence\testTraitFormat.java

**public** **class** AutoTest {

**public** **static** **void** testTraitFormat(String fichierJdt, String fichierRun){

**try**{

//ouverture du jdt

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(fichierJdt)));

//création du fichier jdt.run

PrintWriter result = **new** PrintWriter(**new** FileWriter(fichierRun));

String lectureLigneFichierTest;

//tant que l'on peut lire le fichier

**while**((lectureLigneFichierTest = br.readLine()) != **null**){

//on récupère le numéro de la ligne et la chaîne hexadécimal

String numLigne = lectureLigneFichierTest.substring(0, 2);

String chaineHexadecimal = lectureLigneFichierTest.substring(4);

//on envoie au stub

Format res = stub.Stub.*TraitdeFormat*(chaineHexadecimal);

//on écrit le résultat dans un fichier

result.println(numLigne + " " + res);

}

//on arrête de lire et d'écrire dans les fichiers

br.close();

result.close();

System.*out*.println("Fichier " + fichierRun + " généré.");

}

**catch**(Exception e){

System.*out*.println(e);

}

}

}

### Les fichiers d’entrée et de sorties

[IN] - Data test\TestTraitFormat.jdt

1 00

2 0000

3 00 00

4 99

[OUT] - Data test \JDT.run

1 F1 0

2 F2 0 0

3 F3 0 0

4 F1 153

[REF] - Data test \JDT.ref

1 F1 0

2 F2 0 0

3 F3 0 0

4 F1 153